Customer Rating Reactions Can Be Predicted Purely Using App Features

Federica Sarro*, Mark Harman*†, Yue Jia*†, Yuanyuan Zhang*
*CREST, Department of Computer Science, University College London, London, UK
†Facebook, London, UK
{f.sarro, mark.harman, yue.jia, yuanyuan.zhang}@ucl.ac.uk {markharman, yuej}@fb.com

***Abstract*—In this paper we provide empirical evidence that the rating that an app attracts can be accurately predicted from the features it offers. Our results, based on an analysis of 11,537 apps from the Samsung Android and BlackBerry World app stores, indicate that the rating of 89% of these apps can be predicted with 100% accuracy. Our prediction model is built by using feature and rating information from the existing apps offered in the App Store and it yields highly accurate rating predictions, using only a few (11-12) existing apps for case-based prediction.**

These findings may have important implications for requirements engineering in app stores: They indicate that app developers may be able to obtain (very accurate) assessments of the customer reaction to their proposed feature sets (requirements), thereby providing new opportunities to support the requirements elicitation process for app developers.

***Index Terms*—App Store Analysis, Requirements Elicitation, App Features Extraction, Rating Estimation, Mobile Applications, Software Analytics, Predictive Modelling, Natural Language Processing, Machine Learning, Case Based Reasoning.**

I. Introduction

App development is an increasingly innovative software industry [1]. However, app developers face competitive markets where good apps can fail, receiving poor attention (i.e., few or no downloads) [2]. According to the analytical firms Adeven, Distimo and Localytics, more than 60% of the apps in the Apple App Store have never been downloaded in 2012 [3] and 80% of paid Android apps received fewer than 100 downloads in 2011 [4], and in 2017 80% of all app users, across all industries, abandoned an app within 90 days, with 25% of them using the app once only [5].

App developers therefore need to better understand their users' requirements to ensure their apps receive a better reaction once deployed into an App Store. Fortunately, the wealth of publicly available and analysable data in App Stores may hold an answer to this requirements elicitation problem: App stores provide a new channel of communication between software developers and their users [6]. Users' feedback on previous releases (released by competitors as well as by the developers themselves) could thus potentially be used to support novel approaches to requirements elicitation.

This paper further investigates this potential requirements elicitation channel. We present results that reveal that the extra communication channels available for App Store developers may, indeed, support new kinds of requirements elicitation process. A number of factors may play a role in determining whether an app becomes successful (see e.g., [7][8][9]). In this paper we focus purely on the features claimed for apps in their descriptions as one driver of the customers' reaction to the app. A recent study has shown the influence of both product descriptions and online product reviews on the download decisions for mobile apps [10]. Past research has also found that ratings and downloads are often very highly correlated [11] and that rating is one of the key determinants of users' app purchase decisions [12]. These results hold out the hope that app developers might, somehow, use claimed features of existing apps as a form of requirements elicitation for new proposed apps.

In this paper, we present empirical evidence that an initial, coarse-grained, assessment of customers' likely reaction to a proposed set of features is possible by leveraging information about the features of existing apps. Our results reveal that app developers can mine ratings attracted by existing apps and the features they claim to offer, in order to *predict*, accurately, the likely customer rating reaction to a newly proposed app (based on the set of features proposed for this new app). By thinking of such proposed features as potential requirements, we therefore establish an attractive new *predictive modelling mechanism* to support app requirements elicitation. While the rating that an app attracts remains a relatively coarse-grained aspect of user feedback it is nevertheless, a highly important aspect to app developers, because of its correlation with popularity [11] and, therefore, to the likely commercial success of the apps that developers may seek to deploy into an App Store.

As illustrated in Figure 1, our approach predicts the customer rating reaction to a given app by comparing its proposed feature requirements to the technical features (from now on referred to as claimed features) extracted from the thousands of such app descriptions available in app stores. To mine the claimed features we used the approach proposed by Harman et al. [11], while to estimate the rating of target apps by solely relying on apps' claimed features we used Case Based Reasoning (CBR) [13] as explained in Section II.

We applied this approach to 9,588 and 1,949 apps available from the BlackBerry App World store and the Samsung App (Android) store, respectively. It might surprise the reader to learn that app ratings are predictable just from the claims developers make about their features. This is the key insight of our work: App ratings are surprisingly predictable, at least for the two app stores we studied.

2332-6441/18/$31.00 ©2018 IEEE
DOI 10.1109/RE.2018.00018

RE 2018, Banff, Canada
Research Paper

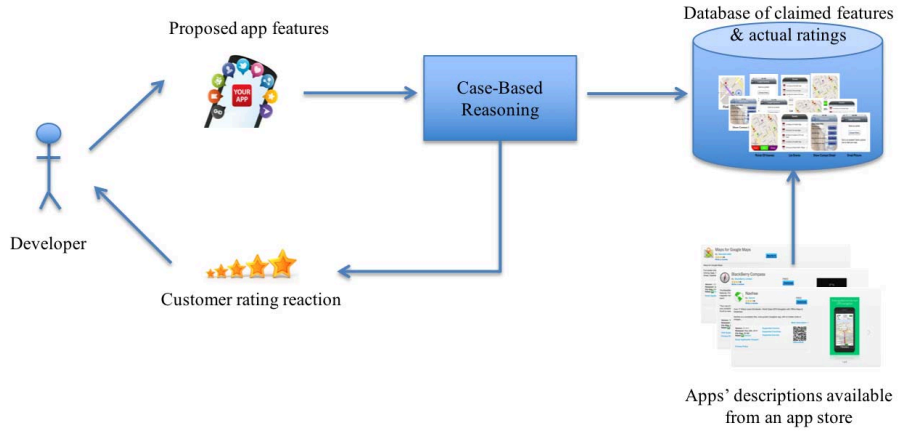


Fig. 1. **Approach Overview**: Estimating the rating of a given app by comparing its proposed features to claimed features and actual ratings of apps available from an app store.

The rest of the paper is organised as follows: Section II describes the approach we used to estimate app ratings solely using information extracted from app stores. Section III describes the design of our empirical study, the results of which are presented in Section IV. Section V describes related work. Section VI provides the reader with actionable findings and future work from our study, while Section VII concludes.

II. Estimating Apps' Rating from Claimed Features

To mine apps' features and rating information from an app store we adopted the app analysis framework proposed by Harman et al. [11]. We used the first three phases to collect feature claims from descriptions and extended the fourth phase to predict rating as follows:

The first phase extracts raw data from the app store (in this study BlackBerry App World[1] and Samsung App[2], though the approach can be applied to other app stores with suitable changes to the extraction front end).

In the second phase, the raw data is parsed to retrieve all the available attributes of each app relating to price, rank of downloads, ratings, and textual descriptions of the app itself.

In the third phase, the framework leverages app descriptions to identify technical information. In particular, it uses Natural Language Processing (NLP) to extract the features of apps from their textual descriptions. The definition of an app feature (as extracted by the Harman et al.'s process [11][14]) is as follows: "A feature is a claimed functionality offered by an app, captured by a set of collocated words in the app description and shared by a set of apps in the same category." For example, "receive Facebook message" and "7-days weather forecast" are two features extracted from apps in the `IM & Social Networking` and `Weather` categories, respectively [14]. In order to extract features from app descriptions written in natural language, a four-step NLP algorithm was devised and implemented by using the Natural Language Toolkit (NLTK), a comprehensive Python package for NLP [15]. In this work we analyse app descriptions written in English. However the approach is language independent and the algorithm can work with different corpora [16]. The details of the algorithm and a running example can be found elsewhere [14].

The fourth and last phase of the approach involves the analysis of the information collected through the previous phases. This phase is application-specific since these information can, of course, support different analyses. In previous work features' metrics have been used to analyse the relationship beetween price, rating, and rank of dowloads [11][14], and claimed features have been used to cluster mobile apps [17]. Sarro et al. [18] have analysed the migration of claimed features across product categories in two existing app stores. In this work we use apps' rating and the features claimed in apps' descriptions as input to the rating prediction system.

Since we aim to study the predictability of apps' rating based only on the claimed features extracted from apps' textual description, we encode the information extracted for a given app as a vector containing the actual app's rating and n technical features extracted from the app's description in the third phase. That is, we form a vector of values that consists of a numerical value representing the rating and a bit string containing one bit per feature that a mobile app may possess: 1 if the app possesses the feature; 0 if not. As an example, if an app resides in a category with 40 potential features, then the vector has 40 bits denoting the presence or absence of these 40 features, and a numeric value denoting its actual rating. The set of all apps collected from existing app stores and encoded as described above forms the database (i.e., case base) we use for our prediction system.

Given a newly proposed, but as-yet unimplemented, app (i.e., characterised purely by its set of proposed features) we retrieve the k most similar apps from this database by using Case Based Reasoning (CBR). CBR is a branch of Artificial Intelligence that has attracted significant interest from a wide range of research domains and it has been successfully used

[1]http://appworld.blackberry.com/webstore/
[2]http://www.samsung.com/levant/apps/mobile/galaxyapps/

in Software Engineering for prediction tasks [19]. Given a target app, CBR identifies similar apps by using a similarity function that measures the distance between the target case and the other cases based on the values of the n features of these apps. Although numerous techniques are available to measure similarity, unweighted Euclidean distance (i.e., the square root of the sum of the squares of the dimension differences) has been the most widely-used in Software Engineering [19], and thus in our empirical study we used the n-dimensional Euclidean distance, where n is the number of features and each app is represented by an n dimensional vector in the space 1^n.

According to the chosen similarity measure, CBR ranks apps in decreasing order of similarity to the target and utilises the past rating of the nearest k apps to estimate the rating of the target app. The choice of k has been a matter of some debate [20]. Earlier studies in Software Engineering restricted their analysis to the closest case (k=1) (see e.g., [21]).More recent studies have used different numbers of analogies to identify the best value for this parameter [22][23][24].

In this work we have analysed the impact of using different number of analogies to shed light on the actionability of CBR for apps' rating prediction. If k is too large then the CBR approach would require too many existing apps' collected data to be actionable. We investigate this actionability concern in RQ3 in the next section.

Once the k most relevant cases have been retrieved, an adaptation strategy is employed to obtain the final estimate for the target app. Many adaptation techniques can be used, based on rules, human expert, or a simple statistical procedure such as a weighted mean [21], which is the measure we used (in this case, the CBR system is also known as k-nearest neighbours).

Further details about the settings used for CBR in our empirical study are given in Section III-D in order to ensure our work and its findings can be replicated by others.

III. Empirical Study Design

This section explains the design of our empirical study, the research questions we set out to answer and the methods and statistical tests we used to answer these questions.

A. Research Questions

Our research questions concern the use of claimed features to predict app ratings based on reasoning by case. If the predictive quality is high, then the obtained estimations may be useful in themselves (to help guide pricing decisions, for example). Such estimations would also validate the quality of the feature information extracted (because they would be extracted to provide accurate predictions of an important attribute: rating). Therefore the first and foremost research question we need to investigate is:

RQ1: Rating Predictability. Can we predict apps' rating from their claimed features?

In order to assess whether rating is predictable using only the claimed features, we compare the CBR approach to Random Guessing (RG), which is a benchmark needed to assess the usefulness of any prediction system [25]. RG randomly assigns the y value of another case to the target case. More formally, it is defined as: Predict a y for the target case t by randomly sampling (with equal probability) over all the remaining $n-1$ cases and take $y = r$ where r is drawn randomly from $1...n^r = t$ [25]. Any good prediction system should outperform random guessing, since an inability to obtain better estimates than random estimates implies that the system is not using any case information [25]. In our case, failure to comfortably outperform RG would imply that the claimed features do not provide CBR with any useful information to predict ratings.

If we find out that the rating is predicable overall, we would still need to understand whether rating predictability varies across categories. It could be that some categories are relatively predictable and this would limit the applicability to app store requirements elicitation. This motivates our second research question:

RQ2: Rating Predictability Across Categories. Does rating predictability vary across app stores' categories?

To answer this question we compare CBR's results to Random Guessing at a finer granularity level, and also compare CBR to two additional baselines based on the use of mean and median value of the ratings of all the apps existing in a given category. Mean and median are two baselines for rating predictability per category (we refer to these approaches as the MeanRating and MedianRating from now onwards). This choice is motivated by the fact that the apps in a certain category are usually grouped based on the functionalities they provide and one may argue that a simple approach such as computing the mean (or median) of the ratings of the existing apps in a given category would give us a good indication for the rating of a new app meant for that category. If so, ratings would not really be predictable, it would simply be that rating distributions highly cluster around the mean/median. Moreover, MeanRating and MedianRating are two common benchmarks for prediction systems when there is no other state-of-the-art approach for comparison as in our case [25]. Therefore, we formulate the following subquestions:

RQ2.1: Does CBR outperform Random Guessing for all categories?

RQ2.2: Does CBR outperform MeanRating for all categories?

RQ2.3: Does CBR outperform MedianRating for all categories?

If we find that rating is predictable within each of the app stores' categories, we need to understand how to use our approach in practice, because our prediction system needs to select the top k similar apps as bases for prediction. This choice is usually left to the developers. Therefore, investigating how robust our approach is to the selection of k provides developers with guidelines for this choice:

RQ3: Actionability. How robust is our approach to the choice of the number of apps, k, used to form the cases for predictive modelling?

In particular, we are interested in observing whether the prediction performance differs significantly depending on the

number of analogies, k, used and whether we can achieve reliable prediction using few cases. If prediction requires too many cases this may make the application to requirements elicitation impractical because it relies on too much information. Therefore, we answer the following subquestions:
RQ3.1: Does CBR provide significantly different predictions depending on the number of apps, k, used?
RQ3.2: Can reliable prediction be achieved with relatively few cases (k)?

In order to answer these questions, we compare the prediction accuracy achieved by CBR exploiting different choices for the number of relevant apps (i.e., from 1 up to 15 analogies) to obtain the final prediction. This will reveal the number of apps that are needed as inputs to the case-based reasoning in order to perform accurate prediction for the stores we considered.

B. Data

To answer the research questions, we constructed an app store database from the information present on the 1st of September 2011 in the BlackBerry App World and Samsung Android App stores for all non-zero priced apps.

Fortunately, the 2011 dataset offers complete data (while the majority of the current app stores only provide information about the highest performing apps). Since we have all the data available in both stores at that time, this study does not suffer from the App Sampling Problem [26]. Of course our technique can be applied to any year, any data set, and any store. However, the ability to overcome the App Sampling Problem ensures that our evaluation is more robust and includes modestly performing apps as well as very popular ones. Our findings therefore hold for all apps in these stores, not merely the most popular.

In September 2011, BlackBerry App World had 19 categories and 9,588 paid apps, while Samsung Android App had 14 categories and 1,949 paid apps. For our analysis we excluded from the BlackBerry study the 'Reference & eBooks' and 'Themes' categories because their apps delivered solely static content (e.g., download books or wallpaper). Moreover, we excluded from the Samsung App study the 'Brand' and 'News & Magazine' categories since they contained very few apps (8 and 12, respectively) and the 'Handmark' category since it contained only apps developed by a same software company (i.e., Handmark). These categories all denote outliers. For the remaining categories, a total of 11,537 apps (9,588 for BlackBerry and 1,949 for Samsung) were available and we extracted 1,256 and 620 features from the BlackBerry apps and the Samsung apps, respectively.

Summary data concerning the categories we studied is presented in Table I: The number of features represents the total number of features extracted from the app descriptions in a given category by using the framework explained in Section II ; The rating data is recorded by the app stores and computed as the average of the ratings given by the customers of a given app (i.e., the rating of an app, as provided by the app stores, is a real number ranging from 0 to 5); we report in the Table the mean rating of the apps per category.

TABLE I
Summary data for the (a) BlackBerry and (b) Samsung stores.

(a) BlackBerry App World

Category	Apps	Features	Rating
Games	2,604	40	2.13
Utilities	1,362	78	2.32
Entertainment	908	86	1.86
Travel	764	81	0.67
Health & Wellness	626	84	1.58
Education	576	82	1.38
Productivity	503	92	2.54
Music & Audio	499	82	0.99
Photo & Video	393	86	1.40
Business	350	96	1.79
Maps & Navigation	245	71	2.16
Sports & Recreation	239	43	2.05
Finance	193	75	1.93
IM & Social Networking	150	78	2.55
News	73	53	1.73
Weather	58	73	2.44
Shopping	45	56	2.33
All categories	9,588	1,256	3.20

(b) Samsung App

Category	Apps	Features	Rating
Entertainment	407	98	1.59
Games	102	72	2.14
Healt/Life	252	53	2.19
Music Video	72	36	1.90
Navigation	130	80	1.93
Productivity	145	87	2.20
References	346	79	0.86
Social	35	25	2.10
Utilities	460	90	1.94
All categories	1,949	620	1.87

We make all our data available to support subsequent downstream analyses[3].

C. Evaluation Criteria and Validation Method

To assess the accuracy of the rating estimations obtained using CBR, we used the most recent best practice for assessing and comparing prediction systems as proposed in previous work [25][27][28].

As suggested by Shepperd and MacDonell [25] we measured the accuracy of a prediction system as the Mean of Absolute Residual (MAR) errors, where the Absolute Residual (AR) error is defined as the absolute value of the difference between the observed value of the dependent variable (y) and the predicted value ($\hat{y}$). In our case the dependent variable is the rating and the AR indicates the prediction error we commit in its prediction.

To check for statistical significance we used the Wilcoxon Signed Rank test [29], since the Shapiro test [30] showed that many of our samples came from non-normally distributed populations, making the t-test unsuitable. The Wilcoxon test is a safe test to apply (even for normally distributed data), since it raises the bar for significance, by making no assumptions about underlying data distributions. In particular, to answer RQ1 and RQ2 we tested the following Null Hypothesis: "*The*

[3]The data will be made publicly available at http://www0.cs.ucl.ac.uk/staff/F.Sarro/projects/UCLappA/home.html.

absolute residual errors provided by the prediction system P_i are not significantly less than those provided by the prediction system P_j.", while to answer RQ3 we tested the Null Hypothesis: "*The absolute residual errors provided by CBRk are not different from those provided by CBR$k+1$*" for $k = 1, ..., 14$. For these tests we set the confidence limit, α, at 0.05 and applied the standard Bonferroni correction (which is the most conservatively cautious of all corrections) to account for multiple statistical hypotheses testing. This conservatism avoids the risk of Type I error (i.e., incorrectly rejecting the Null Hypothesis and claiming predictability without strong evidence).

We also quantified the magnitude of the differences found by using the non-parametric Vargha and Delaney's A_{12} effect size measure [31]. We used this non-parametric measure because not all samples were normally distributed and, as suggested elsewhere [25][32], it is better, in cases such as ours, to use a standardised measure rather than a pooled measure such as the Cohen's d effect size. Given a performance measure M, the A_{12} statistic measures the probability that running algorithm A yields better M-values than running another algorithm B: $A_{12} = (R_1/m - (m+1)/2)/n$, where R_1 is the rank sum of the first data group we are comparing, and m and n are the number of observations in the first and second data sample, respectively. If the two algorithms are equivalent, then A_{12} = 0.5. According to Vargha and Delaney [31], a small, medium, and a large difference between two populations is indicated by A_{12} over 0.56, 0.64, and 0.71, respectively. No transformation of the A_{12} metric is needed as we are interested in *any* improvement in predictive performance [33][34].

To validate our prediction system, we used a standard 10-fold cross-validation, which partitions the initial data set of m observations into 10 randomly-selected test subsets of nearly equal size. For each of the test subsets, the remaining observations compose a training set which is used to build the estimation model; such a model is then validated on the corresponding test subset.

D. Case Based Reasoning Setting

To realise the CBR system used in this study we relied on the ANGEL tool originally developed by Shepperd and Schofield to estimate the development effort of software projects [13]. ANGEL is publicly available and can be easily used for other prediction tasks. It supports the Euclidean distance measure between vectors and we used this metric to compute app similarity, while the final estimation was computed as the mean rating of the k nearest analogies (i.e., the k most similar apps). We report results of each of the choices of k, between $k = 1$ and $k = 15$ analogies. To answer RQs 1-2 we consider the worst, the mean and the best results (given the choice of k), while in Section IV-C we report the analysis of the performance of CBR for each of the k we investigated (i.e., $1 \leq k \leq 15$) to answer RQ3.

E. Threats to Validity

We discuss the validity of our study based on three types of threats: construct, conclusion, and external validity.

Since our data is extracted from the Samsung Android and Blackberry App Store, we are relying on the maintainers of these stores for the reliability of our raw data. Therefore inaccuracies and imprecision in this data may have affected some of our derived data. For example, customers may, for various reasons, disappropriately leave ratings that do not reflect their true opinions [14] or the current store-rating of apps may not be sufficiently dynamic to capture the changing user satisfaction levels associated with the evolving nature of apps [35]. Nevertheless, it is not unreasonable to hope that broad observations about whole classes of apps may still prove to be robust [14]. Thus, in order to protect against possibly incorrect conclusions drawn from analysing such data, we have been careful to draw all of our primary conclusions from analyses based on large sets of data.

We also addressed possible construct validity threats due to a biased selection of the apps used in empirical studies (a.k.a. the App Sampling Problem [26]) by using all the apps that were present in both stores at the time of collection. This ensures that our evaluation includes modestly performing apps as well as very popular ones. However, our findings may not generalise to those stores that probably suffer the App Sampling Problem such that insufficient data is available.

In order to mitigate conclusion validity threats, we carefully applied the statistical tests by verifying that all the required assumptions are met by the distributions tested, and correcting for multiple statistical hypotheses testing.

Our approach to external threats follows widely recommended "best practice" for empirical Software Engineering. That is, we investigated two different app stores and we studied a wide set of categories to imbue our study with a high degree of diversity in application type, domain and size. Nevertheless we cannot claim that our results generalise beyond the subjects studied. However, we described in detail the approach proposed and the empirical methodology we followed in order to allow other researcher to replicate and extend our work. Moreover, great care is required in extending our findings from 'claimed features' to features that are truly available to users of the app as developers may not be entirely truthful (either intentionally or unintentionally) about the technical claims made [14]. Therefore a potential threat to generalisability may lie in the extraction of feature information from these descriptions [11][14]. This threat has been mitigated by extracting the features from a large and varied collection of app descriptions, and clarifying that it is clearly a constraint of the Harman et al. method [11] as well as of most NLP-based approaches [36]. Therefore, we cannot and do not claim that the extracted features include all the real features of the app, but we can certainly claim that there is evidence (as indicated by a previous human sanity check [37] and previous work [11][14][17][38][39]) that what we mine from apps' descriptions are meaningful

feature descriptions and that they denote features *claimed* to be included in the apps according to the developers' own description. Moreover, previous studies have shown that it is possible to extract features from product descriptions available on-line [36][40][41][42][43] and the use of text written in natural language is growing for requirements elicitation [18][44][45][46][47], software maintenance and evolution [48][49][50][51] and software testing [52][53][54][55].

IV. Empirical Study Results

This section presents the results of our empirical study in answer to the research questions described in Section III.

A. RQ1. Rating is Predictable

Figures 2(a) and 2(b) show the number of BlackBerry and Samsung apps, respectively, for which CBR (configured with the worst, mean and best k) achieved an Absolute Residual (AR) error ranging from 0 to 5. Observe the very large number of apps with $AR = 0$ and the very low one with $AR = 5$, this means that our approach produced most of the time totally accurate estimates (i.e., the actual rating and the estimated one coincide) and very rarely totally inaccurate ones (i.e., the actual rating is 0 and our approach estimates a 5 rating, or vice versa). In particular, the CBR using the worst k and the CBR using the best k achieved $AR = 0$ for 7,143 and 7,750 of the 9,588 total apps contained in the BlackBerry store, respectively, meaning that our approach achieved completely correct prediction for 74% and 81% of the total apps in the worst and best case, respectively. While, the number of apps for which CBR obtained an $AR = 5$ is extremely low (i.e., 0.2% for the worst-performing k and 0.4% for the best k). For the 2,958 apps from the Samsung store, CBR totally correctly predicted the rating of 2,344 apps (79%) using CBR worst k and produced totally inaccurate predictions for only 3% of these apps; while using CBR best k produced totally accurate predictions for 2,861 Samsung apps (97%) and never produced totally inaccurate ones. These results suggest that the app rating is predictable relying solely on claimed features extracted from app descriptions. This approach has allowed us to achieve estimates totally accurate for 89% (CBR best k) and 77% (CBR worst k) of the apps contained in both stores (i.e., 11,537 apps), and the reaming estimates are totally inaccurate only for 0.4% (CBR best k) and $<$1.7% (CBR worst k) apps.

To assess whether these results were not merely due to pure chance, we compared our predictions to random predictions. Figures 3(a) and 3(b) show the boxplots of the Absolute Residual (AR) errors achieved using CBR and Random Guessing to predict the rating of BlackBerry apps and Samsung apps, respectively. Each of the boxplots displays the full range of variation (from min to max), the likely range of variation (i.e., the interquartile range or IQR), the median (depicted with a black horizontal line) and the mean (depicted with a blue diamond) of the distribution considered. In particular, we report the distributions of the Absolute Residual errors obtained for 1,000 Random Guessing trials, CBR with the best and worst number of analogies, and the average CBR results obtained using 1 to 15 analogies. We can observe that the AR errors provided by CBR are on average (mean) lower than 0.5 and much lower than those provided by Random Guessing (which mean AR is above 2) for both the BlackBerry and Samsung stores. The statistical inference analysis we carried out confirms this observation (see Table II): The absolute residual errors provided by CBR are significantly lower than those achieved by Random Guessing always with large effect size for both the stores in 100% of the cases.

TABLE II
RQ1: Comparing CBR (worst, mean and best k) to Random Guessing (RG) using Wilcoxon test on Absolute Residual errors.

App Store	CBR (worst k) vs. RG p-value (A_{12})	CBR (mean k) vs. RG p-value (A_{12})	CBR (best k) vs. RG p-value (A_{12})
BlackBerry	$<$0.001 (0.93)	$<$0.001 (0.96)	$<$0.001 (0.97)
Samsung	$<$0.001 (0.90)	$<$0.001 (0.95)	$<$0.001 (1.00)

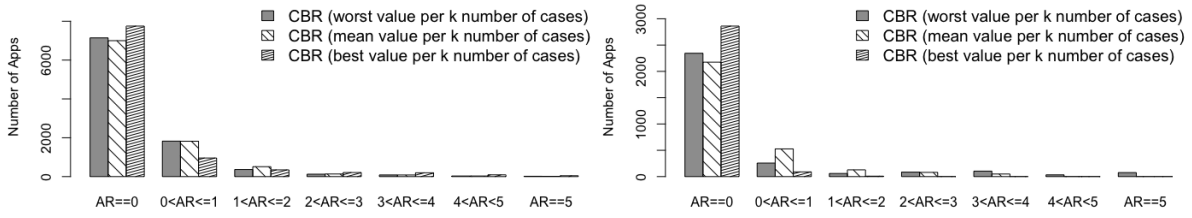
These results allow us to positively answer our first research question **RQ1: Apps' rating is predictable from claimed features for both the BlackBerry and Samsung stores**.

B. RQ2. Predictability Across Categories

Figure 4 reports the Mean of Absolute Residual (MAR) errors obtained by CBR, RandomGuessing, MeanRating and MedianRating for each of the categories of the BlackBerry and Samsung stores. We observe that CBR (mean and best k) achieved the lowest MAR error for all the BlackBerry and Samsung categories indicating that it provided more accurate estimations with respect to RandomGuessing (RQ2.1), MeanRating (RQ2.2) and MedianRating (RQ2.3) for both stores. Moreover, also CBR worst k achieved always a lower MAR than RandomGuessing, and a lower MAR than MeanRating and MedianRating for all categories but one (`Health/Life`).

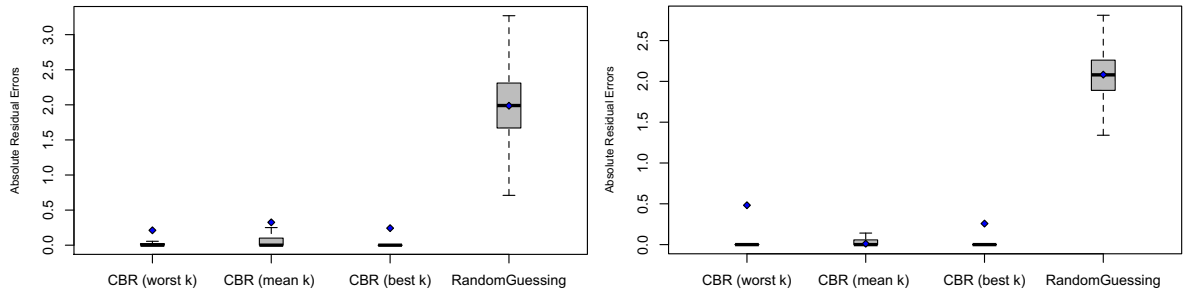
These results are confirmed by the Wilcoxon test, performed on the Absolute Residual (AR) errors provided by the approaches considered. Indeed, the p-values (corrected for multiple statistical testing) and effect sizes reported in Table III for the BlackBerry store show always a significantly lower error of CBR with respect to Random Guessing and with a large effect size for 48 out of 51 cases (94%) and a medium effect size for the remaining three. For the Samsung store, CBR always obtained significantly lower absolute residual errors than Random Guessing with a large effect size for 25 out of 27 cases (93%) and a medium effect size for the remainder. These results allow us to positively answer RQ2.1.

The Wilcoxon test confirms also that, for the BlackBerry store, CBR achieved AR errors significantly better (i.e., lower) than those provided by MeanEffort and MedianEffort in 100 out of 102 cases (98%), while no statistically significant difference was found in the remaining two cases (i.e., CBR-worst-k vs. Mean and vs. MedianRating for the `Weather` category). CBR outperformed MeanRating with large effect sizes in 47 out of 50 cases and medium effect sizes in the remaining cases, while CBR outperformed MedianRating with large (32 out 50 cases) and medium (19 out 50 cases) effect sizes. For the Samsung store, CBR obtained significantly better



(a) BlackBerry App World (b) Samsung App

Fig. 2. **RQ1:** Number of apps with Absolute Residual (AR) errors ranging from 0 to 5 obtained by CBR (worst, mean and best k) for the BlackBerry App World (a) and Samsung App (b) stores.



(a) Prediction Error for BlackBerry App World (b) Prediction Error for Samsung App

Fig. 3. **RQ1:** Boxplots of the Absolute Residual (AR) errors obtained by CBR (worst, mean and best k) and Random Guessing for all the apps of the BlackBerry App World (a) and Samsung App (b) stores (outliers are not visualised for the sake of readability; the mean AR is depicted with a blue diamond).

(i.e., lower) AR errors than those provided by MeanRating and MedianRating in 52 out of 54 cases. In particular, CBR outperformed MeanRating with large effect sizes in all cases, while CBR outperformed MedianRating with large effect size in 22 out 27 cases, medium effect size in one case and small in the remaining four cases. These results allow us to positively answer RQ2.2 and RQ2.3.

In summary, the answer to RQ2 is that apps' rating is predictable across the categories of both app stores in 98% of cases, with large (88%), medium (10%) and small (2%) effect sizes.

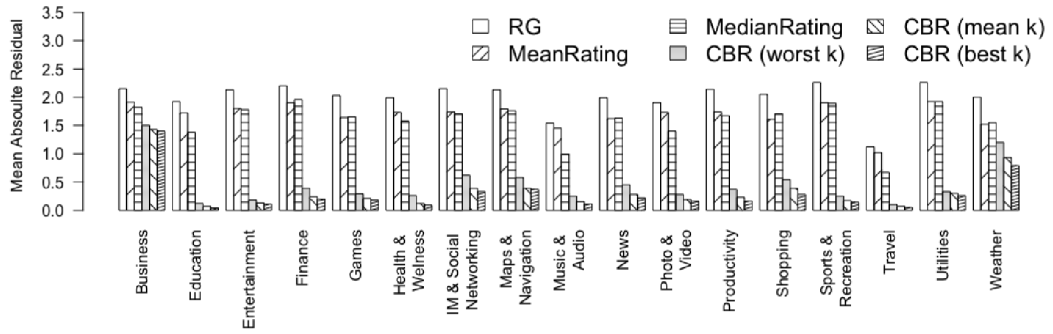
C. RQ3. Actionability

Table IV reports the results in terms of Mean Absolute Residual (MAR) errors obtained with CBR using 1 to 15 analogies (denoted as CBRk, where k is the number of analogies). This allows us to investigate the sensitivity of our approach to this parameter (RQ3.1) and also to determine a suitable recommendation for the value to be used by app developers (RQ3.2) if they want to use our prediction modelling approach to predict ratings.

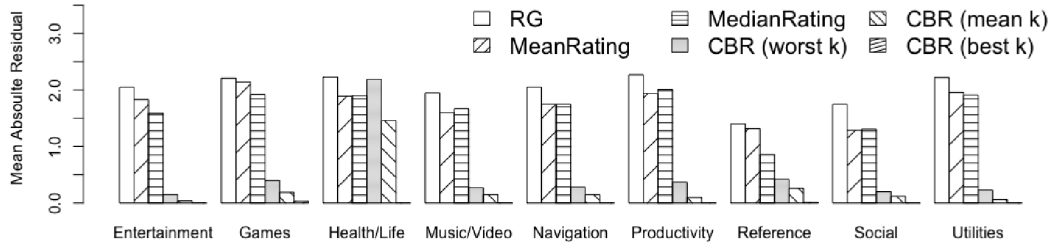
In general, we observe a very low variation in the MAR errors obtained from different numbers of analogies, that is there is a low standard deviation (i.e., < 0.1 for 19 categories, < 0.2 for 5 categories, and ≈ 1 for only 1 category). To further assess whether the prediction performances differ significantly depending on the number of apps, k, used we applied the Wilcoxon test to compare the absolute residual errors obtained by each of the 15 configurations considered. The results revealed a significant difference for only 1,180 out of 5,670 cases (21%). These results suggest that the approach is rather robust to the choice of k (RQ3.1).

In order to investigate whether we can achieve reliable prediction using few cases (RQ3.2), we anlayse the results of the Wilcoxon test according to the following win-tie-loss procedure as in previous work [28][56]: If the distribution i was statistically significantly different than j according to the Wilcoxon test (i.e., $p - value < 0.05/\alpha$, where α is the number of comparisons) we incremented win_i and $loss_j$, otherwise we incremented tie_i and tie_j. Figure 5 shows the percentage of win-tie-loss values for each of the CBRk to illustrates graphically the difference in relative performance of each of the choices for k. From Figure 5(a) (BlackBerry store) we can observe that all CBR configurations using k equals from 4 to 15 provide very similar results, which are only few times better than the configuration using k equals to 1,2, or 3. While, for the Samsung Store (Figure 5(b)) using CBR with 11 to 15 analogies provides often better or similar results with respect to the other choices of k. We also observe that using 11–12 analogies is sufficient to achieve a good win-tie-loss balance for both the Samsung store (83-57-0) and the BlackBerry store (8-229-0),therefore giving practical guidance to the developers who want to apply our approach.

Thus, in answer to RQ3: The approach is robust to different CBR configurations as we found statistically significant differences only for 21% of the comparisons performed for both the BlackBerry and Samsung stores.



(a) BlackBerry App World

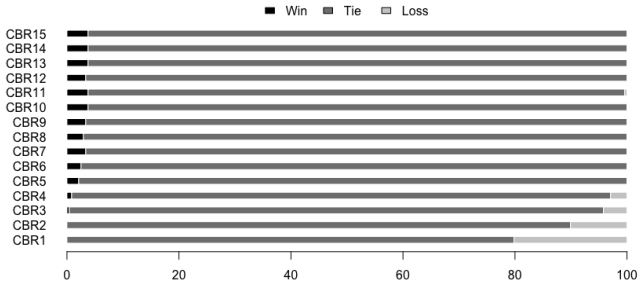


(b) Samsung App

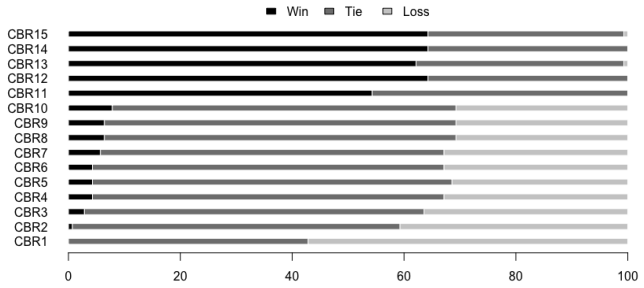
Fig. 4. **RQ2:** Comparing the Mean of Absolute Residual (MAR) errors provided by CBR (worst, mean and best k) and Random Guessing (RG), MeanRating, and MedianRating for each of the categories of the BlackBerry App World (a) and Samsung App stores.

TABLE III
RQ2: Comparing the Absolute Residual (AR) errors of CBR (worst, mean and best k) to Random Guessing, MeanRating and MedianRating, using the Wilcoxon test (p-value (A_{12} effect size)) for the BlackBerry and Samsung app stores.

BlackBerry App World									
Category	CBR (worst k) vs. MeanRating	CBR (mean k) vs. MeanRating	CBR (best k) vs. MeanRating	CBR (worst k) vs. MedianRating	CBR (mean k) vs. MedianRating	CBR (best k) vs. MedianRating	CBR (worst k) vs. RG	CBR (mean k) vs. RG	CBR (best k) vs. RG
Business	<0.001 (0.63)	<0.001 (0.65)	<0.001 (0.65)	<0.001 (0.58)	<0.001 (0.57)	<0.001 (0.58)	<0.001 (0.64)	<0.001 (0.67)	<0.001 (0.67)
Education	<0.001 (0.97)	<0.001 (0.98)	<0.001 (0.99)	<0.001 (0.68)	<0.001 (0.67)	<0.001 (0.67)	<0.001 (0.97)	<0.001 (0.99)	<0.001 (1.00)
Entertainment	<0.001 (0.96)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.93)	<0.001 (0.95)	<0.001 (0.95)	<0.001 (0.97)	<0.001 (0.98)	<0.001 (0.97)
Finance	<0.001 (0.91)	<0.001 (0.96)	<0.001 (0.96)	<0.001 (0.90)	<0.001 (0.95)	<0.001 (0.96)	<0.001 (0.93)	<0.001 (0.98)	<0.001 (0.97)
Games	<0.001 (0.92)	<0.001 (0.94)	<0.001 (0.95)	<0.001 (0.90)	<0.001 (0.92)	<0.001 (0.93)	<0.001 (0.95)	<0.001 (0.99)	<0.001 (0.99)
Health & Wellness	<0.001 (0.94)	<0.001 (0.98)	<0.001 (0.98)	<0.001 (0.69)	<0.001 (0.68)	<0.001 (0.69)	<0.001 (0.94)	<0.001 (0.99)	<0.001 (0.99)
IM & Social Netw.	<0.001 (0.84)	<0.001 (0.91)	<0.001 (0.92)	<0.001 (0.80)	<0.001 (0.87)	<0.001 (0.88)	<0.001 (0.88)	<0.001 (0.98)	<0.001 (0.98)
Maps & Nav.	<0.001 (0.86)	<0.001 (0.92)	<0.001 (0.93)	<0.001 (0.83)	<0.001 (0.89)	<0.001 (0.90)	<0.001 (0.90)	<0.001 (0.97)	<0.001 (0.97)
Music & Audio	<0.001 (0.93)	<0.001 (0.95)	<0.001 (0.97)	<0.001 (0.58)	<0.001 (0.58)	<0.001 (0.58)	<0.001 (0.93)	<0.001 (0.96)	<0.001 (0.97)
News	<0.001 (0.88)	<0.001 (0.94)	<0.001 (0.95)	<0.001 (0.85)	<0.001 (0.92)	<0.001 (0.93)	<0.001 (0.91)	<0.001 (0.99)	<0.001 (0.99)
Photo & Video	<0.001 (0.93)	<0.001 (0.96)	<0.001 (0.96)	<0.001 (0.63)	<0.001 (0.62)	<0.001 (0.62)	<0.001 (0.94)	<0.001 (0.97)	<0.001 (0.97)
Productivity	<0.001 (0.90)	<0.001 (0.94)	<0.001 (0.97)	<0.001 (0.86)	<0.001 (0.86)	<0.001 (0.91)	<0.001 (0.94)	<0.001 (0.99)	<0.001 (1.00)
Shopping	<0.001 (0.84)	<0.001 (0.88)	<0.001 (0.91)	<0.001 (0.85)	<0.001 (0.90)	<0.001 (0.93)	<0.001 (0.94)	<0.001 (0.99)	<0.001 (1.00)
Sports & Recr.	<0.001 (0.94)	<0.001 (0.96)	<0.001 (0.97)	<0.001 (0.93)	<0.001 (0.95)	<0.001 (0.95)	<0.001 (0.97)	<0.001 (1.00)	<0.001 (0.99)
Travel	<0.001 (0.96)	<0.001 (0.97)	<0.001 (0.97)	<0.001 (0.59)	<0.001 (0.60)	<0.001 (0.59)	<0.001 (0.96)	<0.001 (0.97)	<0.001 (0.98)
Utilities	<0.001 (0.93)	<0.001 (0.94)	<0.001 (0.95)	<0.001 (0.90)	<0.001 (0.91)	<0.001 (0.92)	<0.001 (0.95)	<0.001 (0.97)	<0.001 (0.97)
Weather	0.002 (0.62)	<0.001 (0.70)	<0.001 (0.74)	0.003 (0.62)	<0.001 (0.70)	<0.001 (0.75)	<0.001 (0.75)	<0.001 (0.88)	<0.001 (0.88)
Samsung App									
Category	CBR (worst k) vs. MeanRating	CBR (mean k) vs. MeanRating	CBR (best k) vs. MeanRating	CBR (worst k) vs. MedianRating	CBR (mean k) vs. MedianRating	CBR (best k) vs. MedianRating	CBR (worst k) vs. RG	CBR (mean k) vs. RG	CBR (best k) vs. RG
Entertainment	<0.001 (0.96)	<0.001 (0.99)	<0.001 (1.00)	<0.001 (0.69)	<0.001 (0.70)	<0.001 (0.72)	<0.001 (0.96)	<0.001 (0.99)	<0.001 (1.00)
Games	<0.001 (0.72)	<0.001 (0.72)	<0.001 (0.77)	<0.001 (0.98)	<0.001 (0.95)	<0.001 (0.98)	<0.001 (1.00)	<0.001 (1.00)	<0.001 (1.00)
Health/Life	0.70 (0.46)	<0.001 (0.58)	<0.001 (1.00)	0.65 (0.46)	<0.001 (0.60)	<0.001 (0.97)	0.004 (0.47)	<0.001 (0.62)	<0.001 (1.00)
Music/Video	<0.001 (0.92)	<0.001 (0.97)	<0.001 (1.00)	<0.001 (0.91)	<0.001 (0.95)	<0.001 (0.98)	<0.001 (0.95)	<0.001 (0.99)	<0.001 (1.00)
Navigation	<0.001 (0.92)	<0.001 (0.97)	<0.001 (1.00)	<0.001 (0.90)	<0.001 (0.94)	<0.001 (0.97)	<0.001 (0.93)	<0.001 (0.99)	<0.001 (1.00)
Productivity	<0.001 (0.91)	<0.001 (0.99)	<0.001 (1.00)	<0.001 (0.92)	<0.001 (0.98)	<0.001 (1.00)	<0.001 (0.93)	<0.001 (1.00)	<0.001 (1.00)
Reference	<0.001 (0.87)	<0.001 (0.92)	<0.001 (1.00)	<0.001 (0.55)	<0.001 (0.51)	<0.001 (0.60)	<0.001 (0.88)	<0.001 (0.92)	<0.001 (1.00)
Social	<0.001 (0.92)	<0.001 (0.95)	<0.001 (1.00)	<0.001 (0.91)	<0.001 (0.94)	<0.001 (0.99)	<0.001 (0.96)	<0.001 (1.00)	<0.001 (1.00)
Utilities	<0.001 (0.95)	<0.001 (0.99)	<0.001 (1.00)	<0.001 (0.93)	<0.001 (0.98)	<0.001 (0.98)	<0.001 (0.96)	<0.001 (1.00)	<0.001 (1.00)



(a) BlackBerry App World: Comparing CBR settings



(b) Samsung App: Comparing CBR settings

Fig. 5. **RQ3:** Percentage of win-tie-loss obtained from the Wilcoxon test performed on the Absolute Residual errors obtained by using CBR with different numbers of analogies for all the considered categories of the BlackBerry App World (a) and Samsung App (b) stores.

We also found that the use of 11-12 analogies is sufficient to reliably predict the rating across all categories for both the stores.

V. Related Work

In this section we discuss previous work that used the information available in app stores for estimation purposes. A comprehensive literature review of other work on App Store Analysis for Software Engineering can be found elsewhere [6].

No study has been conducted so far on the predictability of rating for mobile apps. However, previous study have found statistical relationships between apps' rating and factors such as number of downloads [11][14], change and fault proneness of Android APIs [7], complexity of user interface [57], application churn [58], bug fixes [9][59], advertisements [60][8] and app size [8][61]. Despite being correlated with ratings, these factors cannot be known at requirements elicitation time for a newly proposed app. Furthermore they are not accessible for competitors' apps. So none of them can be used to estimate the rating that might be accorded to a new app in the early phases of app development such as at requirements elicitation time. Our approach, instead, uses only and solely information publicly available in app stores, which are easy to access and collect, and can be used as a good basis for accurate predictions as shown by our empirical study.

Although no study has focused on rating estimation, a few studies have looked at other estimation tasks, such as estimating mobile apps' size [62][63], cost [64], development effort [65], and crashes [66], as detailed in the following.

Several functional size measurement approaches have been proposed in literature to derive the functional size of mobile apps (e.g., [67][68][69]) and have been used in subsequent studies to estimate the cost and size of mobile apps. Preuss [64] carried out a preliminary case study using Function Points to estimate the cost of an Android app, while Ferrucci et al. [62][63] applied two different COSMIC measurement approaches to 13 Android apps, showing that the apps' functional size was strongly correlated with the final apps' size, and that the functional size could be used to accurately estimate the apps' bytecode size.

Francese et al. [65] investigated prediction models based on information extracted from requirements specification documents (e.g., number of actors, number of use cases) in order to estimate the effort needed to implement an app. They validated this proposal building regression-based estimation models for 23 Android apps and comparing the results to a prediction model based on source code measures (e.g., number of classes, number of files). Their results suggest that the measures from the artefacts produced during the requirements phase are not worse predictors than those based on code source measures.

Xia et at. [66] used machine learning to predict crashing releases (i.e., app releases that are more likely to cause crashes). They collected and used a number of release factors grouped into six categories (complexity, time, code, diffusion, commit, text) and used a Naive Bayes classifier to perform the prediction for 10 open source apps (for a total of 2,638 releases) from the F-Droid repository. Their results revealed that they can improve over a baseline (random) predictor by 50% and 28% in terms of F1 and AUC, respectively.

Finally, several analytic companies (e.g., AppAnnie [70], County [71]) provide usage data to developers, such as downloads, in-app purchase info, etc. Using such information, developers can make data-driven decisions that may allow for more successful apps. However, as observed by Nagappan and Shihab [72], these recommendations are mainly proposed from a marketing perspective rather than a software engineering one.

Of all this previous work, none has provided assistance to app developers in determining arguably the most critical aspect: the app's requirements. Our findings indicate that we can fill this gap using the features previously highly rated by users, as reflected in their predicted rating for previously published apps containing these features.

VI. Discussion and Future Work

Our findings (RQ1 and RQ2) provide evidence to suggest that the combination of app features extraction via NLP and machine learning introduced in this paper can allow us to accurately predict rating. Also, our rating prediction system is based on the simple (and intuitive) approach of case based reasoning. Unlike other more complex machine learning techniques, case based reasoning has relatively few parameters that require tuning. RQ3 shows that 11 or 12 analogies are sufficient for accurately predicting ratings for all categories of

TABLE IV
RQ3: Mean Absolute Residual (MAR) errors obtained with CBR using 1 to 15 analogies to estimate rating for (a) BlackBerry and (b) Samsung apps.

BlackBerry App World																			
Category	CBR1	CBR2	CBR3	CBR4	CBR5	CBR6	CBR7	CBR8	CBR9	CBR10	CBR11	CBR12	CBR13	CBR14	CBR15	Min	Avg	Max	St.Dev.
Business	1.50	1.47	1.48	1.42	1.42	1.40	1.41	1.42	1.41	1.42	1.44	1.43	1.41	1.41	1.41	1.40	1.43	1.50	0.03
Education	0.12	0.09	0.09	0.09	0.07	0.07	0.07	0.06	0.06	0.05	0.04	0.05	0.05	0.05	0.05	0.04	0.07	0.12	0.02
Entertainment	0.18	0.15	0.15	0.13	0.14	0.14	0.12	0.12	0.12	0.12	0.11	0.11	0.11	0.11	0.11	0.11	0.13	0.18	0.02
Finance	0.39	0.35	0.27	0.28	0.24	0.22	0.22	0.21	0.21	0.20	0.20	0.21	0.21	0.21	0.21	0.20	0.24	0.39	0.06
Games	0.29	0.25	0.23	0.21	0.20	0.20	0.19	0.20	0.20	0.19	0.19	0.19	0.19	0.18	0.18	0.18	0.21	0.29	0.03
Health & Wellness	0.26	0.17	0.14	0.12	0.10	0.09	0.11	0.11	0.10	0.10	0.11	0.11	0.12	0.11	0.12	0.09	0.13	0.26	0.04
IM & Social Netw.	0.62	0.55	0.46	0.42	0.39	0.37	0.33	0.34	0.33	0.33	0.34	0.35	0.36	0.36	0.36	0.33	0.39	0.62	0.09
Maps & Nav.	0.58	0.47	0.44	0.40	0.40	0.39	0.39	0.38	0.37	0.37	0.37	0.37	0.38	0.37	0.38	0.37	0.40	0.58	0.06
Music & Audio	0.25	0.17	0.17	0.16	0.17	0.15	0.14	0.14	0.13	0.13	0.13	0.12	0.12	0.11	0.12	0.11	0.15	0.25	0.04
News	0.45	0.39	0.38	0.31	0.28	0.24	0.25	0.23	0.23	0.22	0.22	0.22	0.23	0.25	0.25	0.22	0.28	0.45	0.07
Photo & Video	0.28	0.19	0.18	0.19	0.19	0.20	0.20	0.19	0.19	0.18	0.17	0.17	0.18	0.17	0.16	0.16	0.19	0.28	0.03
Productivity	0.37	0.36	0.31	0.28	0.23	0.21	0.21	0.22	0.20	0.20	0.19	0.18	0.17	0.16	0.16	0.16	0.23	0.37	0.07
Shopping	0.43	0.49	0.48	0.32	0.32	0.28	0.32	0.33	0.29	0.31	0.40	0.45	0.47	0.47	0.54	0.28	0.39	0.54	0.09
Sports & Recr.	0.25	0.24	0.18	0.17	0.14	0.15	0.15	0.14	0.15	0.15	0.15	0.16	0.15	0.15	0.15	0.14	0.17	0.25	0.03
Travel	0.10	0.07	0.08	0.08	0.07	0.07	0.07	0.07	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.07	0.10	0.01
Utilities	0.33	0.27	0.26	0.26	0.26	0.27	0.27	0.26	0.27	0.27	0.28	0.28	0.29	0.29	0.30	0.26	0.28	0.33	0.02
Weather	0.91	0.89	0.79	0.79	0.87	0.83	0.85	0.87	0.90	0.97	0.94	0.97	1.00	1.10	1.20	0.79	0.93	1.20	0.11
Samsung App																			
Category	CBR1	CBR2	CBR3	CBR4	CBR5	CBR6	CBR7	CBR8	CBR9	CBR10	CBR11	CBR12	CBR13	CBR14	CBR15	Min	Avg	Max	St.Dev.
Entertainment	0.15	0.08	0.06	0.05	0.06	0.05	0.05	0.04	0.03	0.03	0.00	0.01	0.00	0.00	0.00	0.00	0.04	0.15	0.04
Games	0.40	0.31	0.29	0.27	0.25	0.27	0.25	0.24	0.23	0.24	0.04	0.04	0.04	0.02	0.03	0.02	0.19	0.40	0.12
Healt/Life	2.18	2.16	2.18	2.17	2.19	2.19	2.19	2.19	2.19	2.19	0.00	0.00	0.00	0.00	0.00	0.00	1.46	2.19	1.07
Music/Video	0.25	0.26	0.26	0.27	0.21	0.22	0.21	0.19	0.18	0.16	0.03	0.01	0.02	0.00	0.01	0.00	0.15	0.27	0.11
Navigation	0.28	0.17	0.19	0.19	0.19	0.20	0.21	0.23	0.23	0.24	0.05	0.02	0.00	0.02	0.00	0.00	0.15	0.28	0.10
Productivity	0.37	0.19	0.13	0.10	0.10	0.11	0.11	0.11	0.09	0.08	0.00	0.01	0.01	0.00	0.01	0.00	0.10	0.37	0.10
Reference	0.42	0.35	0.34	0.39	0.39	0.39	0.40	0.38	0.38	0.38	0.01	0.02	0.02	0.01	0.01	0.01	0.26	0.42	0.18
Social	0.16	0.12	0.13	0.17	0.17	0.17	0.17	0.18	0.19	0.20	0.05	0.02	0.00	0.01	0.00	0.00	0.12	0.20	0.08
Utilities	0.23	0.15	0.10	0.07	0.06	0.06	0.05	0.04	0.04	0.03	0.00	0.00	0.01	0.00	0.00	0.00	0.06	0.23	0.06

both app stores. The implication for app developers is that a prediction system will require mercifully little tuning and can be deployed immediately. Ease of deployment is an important property as developers might not have either the time or the inclination for the tedious and error-prone task of parameter tuning [73].

The scientific evidence we present in this paper is based upon the BlackBerry App World and Samsung Android App markets. Naturally, we cannot (and do not) claim that these findings will necessarily generalise to other markets and platforms. However, our results prospect the possibility of developing suitable prediction systems for other app platforms. The investigation of such systems remains an open problem for future work by the research community.

There also remain many other challenging (but exciting) open problems for App Store Requirement Analysis. For example, a predictive model can be a complement to other requirements elicitation approaches such as user and developer surveys [72][74], or sentiment analysis of customer text reviews [75][76]. Our approach might also help identify features that attract a higher rating, or are unique to certain categories/apps, so that these could be prioritised for development. The prediction of feature popularity may also be useful in prioritising features development and enhancement.

In the future we intend to investigate predictive models of customer evaluations, and the interplay between functional and non-functional properties of apps, and the data available in app stores. For example, the inclusion of other possible success factors and different similarity functions in our proposed CBR system can augment its capabilities to detect interesting cases such as apps having very similar features, but very different user interface or power consumption so that one app is rated higher than the other.

We will also investigate multi-objective predictive models using Search Based Software Engineering (SBSE) [77][78]. The use of multi-objective SBSE, which has provided human-competitive results for other SE prediction tasks [33], can allow us to develop predictive models tailored to the conflicting and competing needs of different app store developers and, perhaps also their customers.

VII. Conclusions

In this paper we have proposed an approach to predict the rating of mobile apps based solely on the technical information gathered from apps' description. We used a Natural Language Processing algorithm to extract, from existing app descriptions, feature information that capture some of the functionalities of these apps in the store [11][14]. This data is the basis for estimates of the rating of apps under consideration at the requirements elicitation phase.

We evaluated our proposed approach on 11,537 apps contained in the BlackBerry App World and Samsung App stores. Our results indicate that app ratings are highly predictable from the claims developers made about their features. The combination of apps' features extraction via NLP and Machine Learning introduced in this paper allowed us to predict with 100% accuracy the rating for 89% of the apps contained in the stores we considered. Moreover, our prediction is consistently reliable across all categories of the stores. Finally, the configuration of the case-base prediction system does not dramatically impact on the prediction performance and the best prediction can be obtained in practice using only few apps, making rating prediction based on proposed app requirements readily deployable in practice.

Acknowledgement

This work is partly supported by the ERC advanced grant Evolutionary Program Improvement Collaborators (EPIC) and by the EPSRC programme grant Dynamic Adaptive Automated Software Engineering (DAASE), EP/J017515/1.

REFERENCES

[1] A. Holzer and J. Ondrus, "Mobile application market: A developer's perspective," *Telemat. Inf.*, vol. 28, no. 1, pp. 22–31, Feb. 2011.

[2] S. Lim, P. Bentley, N. Kanakam, F. Ishikawa, and S. Honiden, "Investigating country differences in mobile app user behavior and challenges for software engineering," *Software Engineering, IEEE Transactions on*, vol. 41, no. 1, pp. 40–64, Jan 2015.

[3] Adeven, http://techcrunch.com/tag/adeven/, accessed: 2018-02-07.

[4] GSMArena, http://blog.gsmarena.com/80-of-paid-apps-in-android-market-get-downloaded-less-than-100-times/, accessed: 2018-02-07.

[5] Localytics, http://info.localytics.com/blog/mobile-apps-whats-a-good-retention-rate, accessed: 2018-02-07.

[6] W. Martin, F. Sarro, Y. Jia, Y. Zhang, and M. Harman, "A survey of app store analysis for software engineering," *IEEE Transactions on Software Engineering*, vol. 43, no. 9, pp. 817–847, Sept 2017.

[7] G. Bavota, M. Linares-Vasquez, C. Bernal-Cardenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk, "The impact of API change- and fault-proneness on the user ratings of android apps," *IEEE Transactions on Software Engineering*, vol. 41, no. 4, pp. 384–407, 2015.

[8] Y. Tian, M. Nagappan, D. Lo, and A. Hassan, "What are the characteristics of high-rated apps? a case study on free android applications," in *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, 2015, pp. 301–310.

[9] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis for app releases in google play," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2016. New York, NY, USA: ACM, 2016, pp. 435–446.

[10] S. Cheng and B. Meszaros, "The influence of online product reviews on the downloading decision for mobile apps," Dept. of Industrial Economics, Blekinge Institute of Technology, Mater Thesis IY2578, 2015.

[11] M. Harman, Y. Jia, and Y. Zhang, "App Store Mining and Analysis: MSR for App Stores," in *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories (MSR '12)*. Zurich, Swiss: IEEE, June 2012, pp. 108–111.

[12] J. E. S. Hee-Woong Kim, Hyun Lyung Lee, "An exploratory study on the determinants of smartphone app purchase," in *In Proceedings of the 11th International DSI and the 16th APDSI Joint Meeting, Taipei, Taiwan, 2011*, 2011.

[13] M. Shepperd and C. Schofield, "Estimating Software Project Effort using Analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736 –743, Nov 1997.

[14] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Investigating the relationship between price, rating, and popularity in the blackberry world app store," *Information & Software Technology*, vol. 87, pp. 119–139, 2017.

[15] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," in *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics (TeachNLP '02)*. Philadelphia, USA: Association for Computational Linguistics, 7-12 July 2002, pp. 69–72.

[16] E. L. Steven Bird, Ewan Klein, *Natural Language Processing with Python Analyzing Text with the Natural Language Toolkit*. OŔeilly Media, 2009.

[17] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang, "Clustering mobile apps based on mined textual features," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016*, 2016, pp. 38:1–38:10.

[18] F. Sarro, A. A. Al-Subaihin, M. Harman, Y. Jia, W. Martin, and Y. Zhang, "Feature lifecycles as they spread, migrate, remain, and die in app stores," in *23rd IEEE International Requirements Engineering Conference, RE 2015*, 2015, pp. 76–85.

[19] M. Shepperd, "Case-based reasoning and software engineering," in *Managing Software Engineering Knowledge*, A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, Eds. Springer Berlin Heidelberg, 2003, pp. 181–198.

[20] G. Kadoda, M. Cartwright, and M. Shepperd, "Issues on the effective use of cbr technology for software project prediction," in *Case-Based Reasoning Research and Development*, ser. Lecture Notes in Computer Science, D. Aha and I. Watson, Eds. Springer Berlin Heidelberg, 2001, vol. 2080, pp. 276–290.

[21] L. Briand, T. Langley, and I. Wieczorek, "A replicated Assessment and Comparison of Common Software Cost Modeling Techniques," in *Proceedings of the 21st International Conference on Software Engineering (ICSE '99)*. Los Angeles, California, USA: IEEE, 16-22 May 1999, pp. 313–322.

[22] F. Ferrucci, E. Mendes, and F. Sarro, "Web Effort Estimation: The Value of Cross-company Data Set Compared to Single-company Data Set," in *Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE '12)*. Lund, Sweden: ACM, 21-22 September 2012, pp. 29–38.

[23] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Software Engineering*, vol. 14, no. 5, pp. 540–578, October 2009.

[24] J. Huang, J. W. Keung, F. Sarro, Y.-F. Li, Y. Yu, W. Chan, and H. Sun, "Cross-validation based k nearest neighbor imputation for software quality datasets: An empirical study," *Journal of Systems and Software*, vol. 132, pp. 226 – 252, 2017.

[25] M. J. Shepperd and S. G. MacDonell, "Evaluating Prediction Systems in Software Project Estimation," *Information & Software Technology*, vol. 54, no. 8, pp. 820–827, August 2012.

[26] W. Martin, M. Harman, Y. Jia, F. Sarro, and Y. Zhang, "The app sampling problem for app store mining," in *IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015*, 2015, pp. 123–133.

[27] W. B. Langdon, J. J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, MARP0," *Information & Software Technology*, vol. 73, pp. 16–18, 2016.

[28] F. Sarro and A. Petrozziello, "LP4EE: Linear Programming as a Baseline for Software Effort Estimation," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 2018.

[29] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences (2nd Edition)*, 2nd ed. Routledge Academic, Jan. 1988.

[30] J. P. Royston, "An Extension of Shapiro and Wilk's W Test for Normality to Large Samples," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 2, pp. 115–124, 1982.

[31] A. Vargha and H. Delaney, "A critique and improvement of the cl common language effect size statistics of mcgraw and wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[32] A. Arcuri and L. Briand, "A Practical Guide for using Statistical Tests to Assess Randomized Algorithms in Software Engineering," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Hawaii, USA: ACM, 21-28 May 2011, pp. 1–10.

[33] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. ACM, 2016, pp. 619–630.

[34] G. Neumann, M. Harman, and S. Poulding, "Transformed vargha-delaney effect size," in *Search-Based Software Engineering*, M. Barros and Y. Labiche, Eds. Springer International Publishing, 2015, pp. 318–324.

[35] J. I. Mojica Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and E. A. Hassan, "An examination of the current rating system used in mobile app stores," *IEEE Software*, pp. 86–92, 2015.

[36] H. Dumitru, M. Gibiec, N. Hariri, J. Cleland-Huang, B. Mobasher, C. Castro-Herrera, and M. Mirakhorli, "On-demand Feature Recommendations Derived from Mining Public Product Descriptions," in *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Hawaii, USA: ACM, 21-28 May 2011, pp. 181–190.

[37] A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store analysis: Mining app stores for relationships between customer, business and technical characteristics," *UCL - Research Note RN/14/10*, September 2014.

[38] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller, "Checking app behavior against app descriptions," in *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*, 2014, pp. 1025–1035.

[39] A. A. Al-Subaihin, A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "App store mining and analysis," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, DeMobile 2015*, 2015, pp. 1–2.

[40] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," in *Proceedings of the 2013 9th Joint*

Meeting on Foundations of Software Engineering, ser. ESEC/FSE 2013, 2013, pp. 290–300.

[41] N. Hariri, C. Castro-Herrera, M. Mirakhorli, J. Cleland-Huang, and B. Mobasher, "Supporting domain analysis through mining and recommending features from online product listings," *Software Engineering, IEEE Transactions on*, vol. 39, no. 12, pp. 1736–1752, Dec 2013.

[42] T. Johann, C. Stanik, A. M. A. B., and W. Maalej, "Safe: A simple approach for feature extraction from app descriptions and app reviews," in *IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 21–30.

[43] T. Quirchmayr, B. Paech, R. Kohl, and H. Karey, "Semi-automatic software feature-relevant information extraction from natural language user manuals - an approach and practical experience at roche diagnostics gmbh," in *Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings*, 2017, pp. 255–272.

[44] A. Sutcliffe and P. Sawyer, "Requirements elicitation: Towards the unknown unknowns," in *IEEE International Requirements Engineering Conference*, 2013, pp. 92–104.

[45] A. Massey, J. Eisenstein, A. Anton, and P. Swire, "Automated text mining for requirements analysis of policy documents," in *IEEE International Requirements Engineering Conference*, 2013, pp. 4–13.

[46] E. Guzman, R. Alkadhi, and N. Seyff, "A needle in a haystack: What do twitter users say about software?" in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sept 2016, pp. 96–105.

[47] N. Jha and A. Mahmoud, "Mining user requirements from application store reviews using frame semantics," in *Requirements Engineering: Foundation for Software Quality*, P. Grünbacher and A. Perini, Eds. Springer International, 2017, pp. 273–287.

[48] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Extracting domain models from natural-language requirements: Approach and industrial evaluation," in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '16. New York, NY, USA: ACM, 2016, pp. 250–260.

[49] E. Guzman, M. Ibrahim, and M. Glinz, "A little bird told me: Mining tweets for requirements and software evolution," in *25th IEEE International Requirements Engineering Conference, RE 2017, Lisbon, Portugal, September 4-8, 2017*, 2017, pp. 11–20.

[50] ——, "Mining twitter messages for software evolution," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 283–284.

[51] M. Nayebi, H. Cho, H. Farrahi, and G. Ruhe, "App store mining is not enough," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 152–154.

[52] B. Rosadini, A. Ferrari, G. Gori, A. Fantechi, S. Gnesi, I. Trotta, and S. Bacherini, "Using NLP to detect requirements defects: An industrial experience in the railway domain," in *Requirements Engineering: Foundation for Software Quality - 23rd International Working Conference, REFSQ 2017, Essen, Germany, February 27 - March 2, 2017, Proceedings*, 2017, pp. 344–360.

[53] M. Harman, A. Al-Subaihin, Y. Jia, W. Martin, F. Sarro, and Y. Zhang, "Mobile app and app store analysis, testing and optimisation," in *Proceedings of the International Conference on Mobile Software Engineering and Systems*, ser. MOBILESoft '16. ACM, 2016, pp. 243–244.

[54] Y. Zhang, M. Harman, Y. Jia, and F. Sarro, "Inferring test models from kate's bug reports using multi-objective search," in *Search-Based Software Engineering*, M. Barros and Y. Labiche, Eds. Springer International Publishing, 2015, pp. 301–307.

[55] F. Sarro, "Predictive analytics for software testing," in *ACM/IEEE 11th International Workshop on Search-Based Software Testing, (SBST?18) May 28?29, 2018, Gothenburg, Sweden*, 2018, p. 1.

[56] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren, "Adaptive multi-objective evolutionary algorithms for overtime planning in software projects," *IEEE Trans. Software Eng.*, vol. 43, no. 10, pp. 898–917, 2017.

[57] S. E. S. Taba, I. Keivanloo, Y. Zou, J. Ng, and T. Ng, "An exploratory study on the relation between user interface complexity and the perceived quality," in *Web Engineering*, S. Casteleyn, G. Rossi, and M. Winckler, Eds. Springer International, 2014, pp. 370–379.

[58] L. Guerrouj, S. Azad, and P. Rigby, "The influence of app churn on app success and stackoverflow discussions," in *IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER), 2015*, 2015, pp. 321–330.

[59] W. Martin, F. Sarro, and M. Harman, "Causal impact analysis applied to app releases in google play and windows phone store," University College London, Tech. Rep., 2015. [Online]. Available: http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/research/Research_Notes/RN_15_07.pdf

[60] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond, "Truth in advertising: The hidden cost of mobile ads for software developers," in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15, 2015, pp. 100–110.

[61] E. Shaw, A. Shaw, and D. Umphress, "Mining android apps to predict market ratings," in *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on*, 2014, pp. 166–167.

[62] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, "Investigating functional and code size measures for mobile applications," in *41st Euromicro Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2015*, 2015, pp. 365–368.

[63] ——, "Investigating functional and code size measures for mobile applications: A replicated study," in *Proceedings of the 16th International Conference on Product-Focused Software Process Improvement, PROFES 2015*, 2015, pp. 271–287.

[64] T. Preuss, "Mobile applications, function points and cost estimating," in *Proceedings of the International Cost Estimation and Analysis Association Conference*, 2013.

[65] R. Francese, C. Gravino, M. Risi, G. Scanniello, and G. Tortora, "On the use of requirements measures to predict software project and product measures in the context of android mobile apps: A preliminary study," in *41st Euromicro Conference on Software Engineering and Advanced Applications, EUROMICRO-SEAA 2015*, 2015, pp. 357–364.

[66] X. Xia, E. Shihab, Y. Kamei, D. Lo, and X. Wang, "Predicting crashing releases of mobile applications," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2016, Ciudad Real, Spain, September 8-9, 2016*, 2016, pp. 29:1–29:10.

[67] G. Sethumadhavan, "Sizing android mobile applications," in *In Proceedings od tge 6th IFPUG International Software Measurement and Analysis Conference (ISMA)*, 2011.

[68] H. van Heeringen and E. Van Gorp, "Measure the functional size of a mobile app: Using the cosmic functional size measurement method," in *Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2014 Joint Conference of the International Workshop on*, 2014, pp. 11–16.

[69] M. Haoues, A. Sellami, and H. Ben-Abdallah, "A rapid measurement procedure for sizing web and mobile applications based on cosmic fsm method," in *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, ser. IWSM Mensura '17. New York, NY, USA: ACM, 2017, pp. 129–137.

[70] "AppAnnie," https://www.appannie.com, accessed: 2018-02-06.

[71] "County," https://count.ly, accessed: 2018-02-06.

[72] M. Nagappan and E. Shihab, "Future trends in software engineering research for mobile apps," in *In Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2016)*, 2016, pp. 21–32.

[73] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Information & Software Technology*, vol. 76, pp. 135–146, 2016.

[74] M. Nayebi, B. Adams, and G. Ruhe, "Release practices in mobile apps? Users and developers perception," in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Osaka, Japan, March 2016.

[75] E. Guzman and W. Maalej, "How do users like this feature? A fine grained sentiment analysis of app reviews," in *Proceedings of the 22nd IEEE International Requirements Engineering Conference (RE) 2014)*, 2014, pp. 153–162.

[76] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan, "What do mobile app users complain about?" *Software, IEEE*, vol. 32, no. 3, pp. 70–77, 2015.

[77] M. Harman and B. F. Jones, "Search Based Software Engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, December 2001.

[78] M. Harman, "The Relationship between Search Based Software Engineering and Predictive Modeling," in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering (PROMISE '10)*. Timisoara, Romania: ACM, 12-13 September 2010, pp. 1–13.